bounding box

May 19, 2019

0.0.1 Read the cropping dataset

```
In [1]: import keras

        with open('cropping.txt', 'rt') as f: data = f.read().split('\n')[:-1]
        data = [line.split(',') for line in data]
        data = [(p,[(int(coord[i]),int(coord[i+1])) for i in range(0,len(coord),2)]) for p,*coo
        data[0]
```

```
/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the se
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
Out[1]: ('88532e70.jpg',
         [(195, 293),
          (269, 115),
          (868, 158),
          (888, 170),
          (641, 496),
          (512, 546),
          (321, 524)])
```

This dataset contains the location of points on the edge of the fluke for 1200 pictures randomly selected from the Humpback Whale Identification Challenge training set. Points are selected to capture the leftmost and rightmost points, as well as the highest and lowest points. Additional points are added to help determine the fluke bounding box following an affine transformation on the image.

The intent of this dataset is to build a model for locating the whale fluke inside the image. In the context of the Humpback Whale Identification Challenge, such a model can then be used to crop images around the region of interest.

```
In [2]: from PIL import Image as pil_image
        from PIL.ImageDraw import Draw
        from os.path import isfile

        def expand_path(p):
            if isfile('train/' + p): return 'train/' + p
```

```
        if isfile('test/' + p): return 'test/' + p
        return p

    def read_raw_image(p):
        return pil_image.open(expand_path(p))

    def draw_dot(draw, x, y):
        draw.ellipse(((x-5,y-5),(x+5,y+5)), fill='red', outline='red')

    def draw_dots(draw, coordinates):
        for x,y in coordinates: draw_dot(draw, x, y)

    def bounding_rectangle(list):
        x0, y0 = list[0]
        x1, y1 = x0, y0
        for x,y in list[1:]:
            x0 = min(x0, x)
            y0 = min(y0, y)
            x1 = max(x1, x)
            y1 = max(y1, y)
        return x0,y0,x1,y1

    filename,coordinates = data[0]
    box = bounding_rectangle(coordinates)
    img = read_raw_image(filename)
    draw = Draw(img)
    draw_dots(draw, coordinates)
    draw.rectangle(box, outline='red')
    img
```

Out[2]:

In [3]:
```
# Define useful constants
img_shape  = (128,128,1)
anisotropy = 2.15
```

In [4]:
```
import random
import numpy as np
from scipy.ndimage import affine_transform
from keras.preprocessing.image import img_to_array

# Read an image as black&white numpy array
def read_array(p):
    img = read_raw_image(p).convert('L')
    return img_to_array(img)

def build_transform(rotation, shear, height_zoom, width_zoom, height_shift, width_shift
    rotation        = np.deg2rad(rotation)
    shear           = np.deg2rad(shear)
    rotation_matrix = np.array([[np.cos(rotation), np.sin(rotation), 0], [-np.sin(rotat
    shift_matrix    = np.array([[1, 0, height_shift], [0, 1, width_shift], [0, 0, 1]])
    shear_matrix    = np.array([[1, np.sin(shear), 0], [0, np.cos(shear), 0], [0, 0, 1
    zoom_matrix     = np.array([[1.0/height_zoom, 0, 0], [0, 1.0/width_zoom, 0], [0, 0
    shift_matrix    = np.array([[1, 0, -height_shift], [0, 1, -width_shift], [0, 0, 1]
    return np.dot(np.dot(rotation_matrix, shear_matrix), np.dot(zoom_matrix, shift_matr
```

```
# Compute the coordinate transformation required to center the pictures, padding as re
def center_transform(affine, input_shape):
    hi, wi = float(input_shape[0]), float(input_shape[1])
    ho, wo = float(img_shape[0]), float(img_shape[1])
    top, left, bottom, right = 0, 0, hi, wi
    if wi/hi/anisotropy < wo/ho: # input image too narrow, extend width
        w     = hi*wo/ho*anisotropy
        left  = (wi-w)/2
        right = left + w
    else: # input image too wide, extend height
        h      = wi*ho/wo/anisotropy
        top    = (hi-h)/2
        bottom = top + h
    center_matrix   = np.array([[1, 0, -ho/2], [0, 1, -wo/2], [0, 0, 1]])
    scale_matrix    = np.array([[(bottom - top)/ho, 0, 0], [0, (right - left)/wo, 0],
    decenter_matrix = np.array([[1, 0, hi/2], [0, 1, wi/2], [0, 0, 1]])
    return np.dot(np.dot(decenter_matrix, scale_matrix), np.dot(affine, center_matrix))

# Apply an affine transformation to an image represented as a numpy array.
def transform_img(x, affine):
    matrix   = affine[:2,:2]
    offset   = affine[:2,2]
    x        = np.moveaxis(x, -1, 0)
    channels = [affine_transform(channel, matrix, offset, output_shape=img_shape[:-1],
                                 mode='constant', cval=np.average(channel)) for channel
    return np.moveaxis(np.stack(channels, axis=0), 0, -1)

# Read an image for validation, i.e. without data augmentation.
def read_for_validation(p):
    x  = read_array(p)
    t  = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    t  = center_transform(t, x.shape)
    x  = transform_img(x, t)
    x -= np.mean(x, keepdims=True)
    x /= np.std(x, keepdims=True) + K.epsilon()
    return x,t

# Read an image for training, i.e. including a random affine transformation
def read_for_training(p):
    x  = read_array(p)
    t  = build_transform(
            random.uniform(-5, 5),
            random.uniform(-5, 5),
            random.uniform(0.9, 1.0),
            random.uniform(0.9, 1.0),
            random.uniform(-0.05*img_shape[0], 0.05*img_shape[0]),
            random.uniform(-0.05*img_shape[1], 0.05*img_shape[1]))
    t  = center_transform(t, x.shape)
```

```
        x  = transform_img(x, t)
        x -= np.mean(x, keepdims=True)
        x /= np.std(x, keepdims=True) + K.epsilon()
        return x,t

    # Transform corrdinates according to the provided affine transformation
    def coord_transform(list, trans):
        result = []
        for x,y in list:
            y,x,_ = trans.dot([y,x,1]).astype(np.int)
            result.append((x,y))
        return result
```

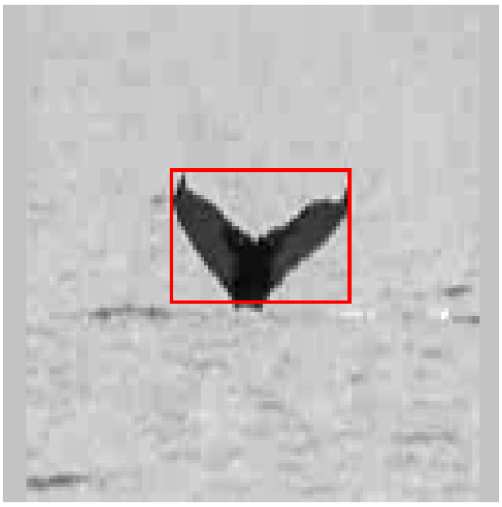
In [5]:
```
from sklearn.model_selection import train_test_split

train, val = train_test_split(data, test_size=200, random_state=1)
train += train
train += train
train += train
train += train
len(train),len(val)
```

Out[5]:
```
(16000, 200)
```

In [7]:
```
import matplotlib.pyplot as plt
from tqdm import tqdm, tqdm_notebook
from keras import backend as K
from keras.preprocessing.image import array_to_img
from numpy.linalg import inv as mat_inv

def show_whale(imgs, per_row=5):
    n         = len(imgs)
    rows      = (n + per_row - 1)//per_row
    cols      = min(per_row, n)
    fig, axes = plt.subplots(rows,cols, figsize=(24//per_row*cols,24//per_row*rows))
    for ax in axes.flatten(): ax.axis('off')
    for i,(img,ax) in enumerate(zip(imgs, axes.flatten())): ax.imshow(img.convert('RGB

val_a = np.zeros((len(val),)+img_shape,dtype=K.floatx()) # Preprocess validation image
val_b = np.zeros((len(val),4),dtype=K.floatx()) # Preprocess bounding boxes
for i,(p,coords) in enumerate(tqdm_notebook(val)):
    img,trans      = read_for_validation(p)
    coords         = coord_transform(coords, mat_inv(trans))
    x0,y0,x1,y1    = bounding_rectangle(coords)
    val_a[i,:,:,:] = img
    val_b[i,0]     = x0
    val_b[i,1]     = y0
    val_b[i,2]     = x1
```

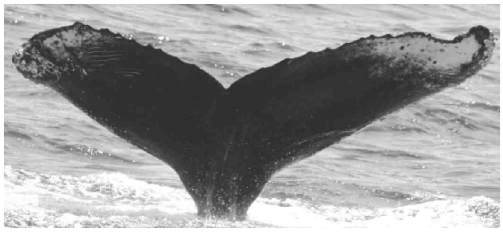
```
            val_b[i,3]     = y1

        idx  = 1
        img  = array_to_img(val_a[idx])
        img  = img.convert('RGB')
        draw = Draw(img)
        draw.rectangle(val_b[idx], outline='red')
        show_whale([read_raw_image(val[idx][0]), img], per_row=2)
```

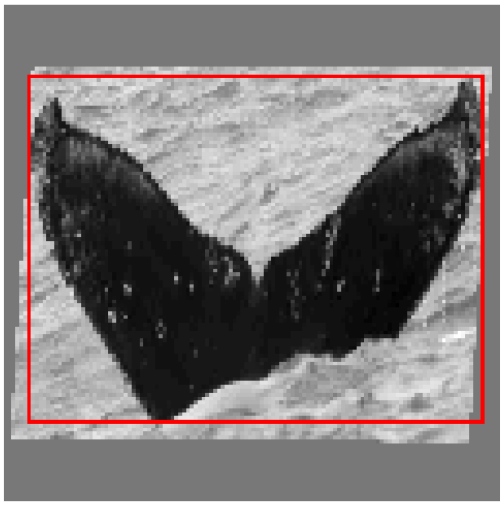
```
HBox(children=(IntProgress(value=0, max=200), HTML(value='')))
```



```
In [8]: from keras.utils import Sequence

        class TrainingData(Sequence):
            def __init__(self, batch_size=32):
                super(TrainingData, self).__init__()
                self.batch_size = batch_size
            def __getitem__(self, index):
                start = self.batch_size*index;
                end   = min(len(train), start + self.batch_size)
                size  = end - start
                a     = np.zeros((size,) + img_shape, dtype=K.floatx())
                b     = np.zeros((size,4), dtype=K.floatx())
                for i,(p,coords) in enumerate(train[start:end]):
                    img,trans   = read_for_training(p)
                    coords      = coord_transform(coords, mat_inv(trans))
```

```
                x0,y0,x1,y1 = bounding_rectangle(coords)
                a[i,:,:,:]  = img
                b[i,0]      = x0
                b[i,1]      = y0
                b[i,2]      = x1
                b[i,3]      = y1
            return a,b
        def __len__(self):
            return (len(train) + self.batch_size - 1)//self.batch_size

    random.seed(1)
    a, b = TrainingData(batch_size=5)[1]
    img  = array_to_img(a[0])
    img  = img.convert('RGB')
    draw = Draw(img)
    draw.rectangle(b[0], outline='red')
    show_whale([read_raw_image(train[0][0]), img], per_row=2)
```





In [9]:
```
from keras.engine.topology import Input
from keras.layers import BatchNormalization, Concatenate, Conv2D, Dense, Dropout, Flat
from keras.models import Model

def build_model(with_dropout=True):
    kwargs     = {'activation':'relu', 'padding':'same'}
    conv_drop  = 0.2
    dense_drop = 0.5
    inp        = Input(shape=img_shape)

    x = inp
```

```
x = Conv2D(64, (9, 9), **kwargs)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = BatchNormalization()(x)
if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, int(x.shape[-1])))

x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = BatchNormalization()(x)
if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, int(x.shape[-1])))

x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = BatchNormalization()(x)
if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, int(x.shape[-1])))

x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = BatchNormalization()(x)
if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, int(x.shape[-1])))

x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = BatchNormalization()(x)
if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, int(x.shape[-1])))

x = Conv2D(64, (2, 2), **kwargs, strides=2)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = Conv2D(64, (3, 3), **kwargs)(x)
x = BatchNormalization()(x)
if with_dropout: x = Dropout(conv_drop, noise_shape=(None, 1, 1, int(x.shape[-1])))

h = MaxPooling2D(pool_size=(1, int(x.shape[2])))(x)
h = Flatten()(h)
if with_dropout: h = Dropout(dense_drop)(h)
h = Dense(16, activation='relu')(h)

v = MaxPooling2D(pool_size=(int(x.shape[1]), 1))(x)
v = Flatten()(v)
if with_dropout: v = Dropout(dense_drop)(v)
v = Dense(16, activation='relu')(v)

x = Concatenate()([h,v])
if with_dropout: x = Dropout(0.5)(x)
x = Dense(4, activation='linear')(x)
```

```
        return Model(inp,x)

    model = build_model(with_dropout=True)
    model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 128, 128, 1)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 64)      5248
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 64)      36928
_________________________________________________________________
batch_normalization_1 (BatchNor (None, 128, 128, 64)   256
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 64)        16448
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 64)        36928
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 64, 64, 64)        36928
_________________________________________________________________
batch_normalization_2 (BatchNor (None, 64, 64, 64)     256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 64)        16448
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
batch_normalization_3 (BatchNor (None, 32, 32, 64)     256
_________________________________________________________________
dropout_3 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 16, 16, 64)        16448
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_4 (BatchNor (None, 16, 16, 64)     256
_________________________________________________________________
dropout_4 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
```

```
conv2d_12 (Conv2D)              (None, 8, 8, 64)     16448       dropout_4[0][0]
____________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 8, 8, 64)     36928       conv2d_12[0][0]
____________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 8, 8, 64)     36928       conv2d_13[0][0]
____________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 8, 8, 64)     256         conv2d_14[0][0]
____________________________________________________________________________________________
dropout_5 (Dropout)             (None, 8, 8, 64)     0           batch_normalization_5[0][0]
____________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 4, 4, 64)     16448       dropout_5[0][0]
____________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 4, 4, 64)     36928       conv2d_15[0][0]
____________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 4, 4, 64)     36928       conv2d_16[0][0]
____________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 4, 4, 64)     256         conv2d_17[0][0]
____________________________________________________________________________________________
dropout_6 (Dropout)             (None, 4, 4, 64)     0           batch_normalization_6[0][0]
____________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 4, 1, 64)     0           dropout_6[0][0]
____________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 1, 4, 64)     0           dropout_6[0][0]
____________________________________________________________________________________________
flatten_1 (Flatten)             (None, 256)          0           max_pooling2d_1[0][0]
____________________________________________________________________________________________
flatten_2 (Flatten)             (None, 256)          0           max_pooling2d_2[0][0]
____________________________________________________________________________________________
dropout_7 (Dropout)             (None, 256)          0           flatten_1[0][0]
____________________________________________________________________________________________
dropout_8 (Dropout)             (None, 256)          0           flatten_2[0][0]
____________________________________________________________________________________________
dense_1 (Dense)                 (None, 16)           4112        dropout_7[0][0]
____________________________________________________________________________________________
dense_2 (Dense)                 (None, 16)           4112        dropout_8[0][0]
____________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 32)           0           dense_1[0][0]
                                                                 dense_2[0][0]
____________________________________________________________________________________________
dropout_9 (Dropout)             (None, 32)           0           concatenate_1[0][0]
____________________________________________________________________________________________
dense_3 (Dense)                 (None, 4)            132         dropout_9[0][0]
============================================================================================
Total params: 503,588
Trainable params: 502,820
Non-trainable params: 768
____________________________________________________________________________________________
```

```
In [10]: from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
         from keras.optimizers import Adam

         for num in range(1, 4):
             model_name = 'cropping-%01d.h5' % num
             print(model_name)
             model.compile(Adam(lr=0.032), loss='mean_squared_error')
             model.fit_generator(
                 TrainingData(), epochs=50, max_queue_size=12, workers=4, verbose=1,
                 validation_data=(val_a, val_b),
                 callbacks=[
                     EarlyStopping(monitor='val_loss', patience=9, min_delta=0.1, verbose=1),
                     ReduceLROnPlateau(monitor='val_loss', patience=3, min_delta=0.1, factor=0
                     ModelCheckpoint(model_name, save_best_only=True, save_weights_only=True),
                 ])
             model.load_weights(model_name)
             model.evaluate(val_a, val_b, verbose=0)

cropping-1.h5
Epoch 1/50
500/500 [==============================] - 2507s 5s/step - loss: 557.0855 - val_loss: 165.9808
Epoch 2/50
500/500 [==============================] - 2540s 5s/step - loss: 346.6362 - val_loss: 120.6931
Epoch 3/50
500/500 [==============================] - 2574s 5s/step - loss: 278.8406 - val_loss: 52731.493
Epoch 4/50
500/500 [==============================] - 2575s 5s/step - loss: 240.7661 - val_loss: 131.0033
Epoch 5/50
500/500 [==============================] - 2531s 5s/step - loss: 208.5899 - val_loss: 12988.347

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.00800000037997961.
Epoch 6/50
500/500 [==============================] - 2482s 5s/step - loss: 179.2290 - val_loss: 68.9442
Epoch 7/50
500/500 [==============================] - 2388s 5s/step - loss: 170.2741 - val_loss: 70.3035
Epoch 8/50
500/500 [==============================] - 2289s 5s/step - loss: 162.0193 - val_loss: 68.0217
Epoch 9/50
500/500 [==============================] - 2284s 5s/step - loss: 152.7652 - val_loss: 56.9733
Epoch 10/50
500/500 [==============================] - 2333s 5s/step - loss: 144.5138 - val_loss: 55.2711
Epoch 11/50
500/500 [==============================] - 2452s 5s/step - loss: 139.8426 - val_loss: 48.1526
Epoch 12/50
500/500 [==============================] - 2447s 5s/step - loss: 131.9710 - val_loss: 54.0475
Epoch 13/50
500/500 [==============================] - 2380s 5s/step - loss: 122.8758 - val_loss: 47.9916
Epoch 14/50
```

```
500/500 [==============================] - 2376s 5s/step - loss: 116.9095 - val_loss: 50.3715
Epoch 15/50
500/500 [==============================] - 2382s 5s/step - loss: 113.2557 - val_loss: 45.1115
Epoch 16/50
500/500 [==============================] - 2382s 5s/step - loss: 105.9497 - val_loss: 40.0823
Epoch 17/50
500/500 [==============================] - 2381s 5s/step - loss: 99.2378 - val_loss: 50.9979
Epoch 18/50
500/500 [==============================] - 2384s 5s/step - loss: 92.9641 - val_loss: 52.6463
Epoch 19/50
500/500 [==============================] - 2376s 5s/step - loss: 89.5002 - val_loss: 40.0454

Epoch 00019: ReduceLROnPlateau reducing learning rate to 0.0020000000949949026.
Epoch 20/50
500/500 [==============================] - 2384s 5s/step - loss: 83.6473 - val_loss: 36.9560
Epoch 21/50
500/500 [==============================] - 2382s 5s/step - loss: 79.4791 - val_loss: 34.4707
Epoch 22/50
500/500 [==============================] - 2383s 5s/step - loss: 79.4619 - val_loss: 30.8307
Epoch 23/50
500/500 [==============================] - 2381s 5s/step - loss: 77.7097 - val_loss: 34.4418
Epoch 24/50
500/500 [==============================] - 2383s 5s/step - loss: 76.6076 - val_loss: 31.3798
Epoch 25/50
500/500 [==============================] - 2361s 5s/step - loss: 75.6522 - val_loss: 32.4307

Epoch 00025: ReduceLROnPlateau reducing learning rate to 0.002.
Epoch 26/50
500/500 [==============================] - 2385s 5s/step - loss: 73.5478 - val_loss: 31.1684
Epoch 27/50
500/500 [==============================] - 2388s 5s/step - loss: 73.4404 - val_loss: 29.9076
Epoch 28/50
500/500 [==============================] - 2385s 5s/step - loss: 71.9389 - val_loss: 32.5809
Epoch 29/50
500/500 [==============================] - 2390s 5s/step - loss: 71.8188 - val_loss: 29.5498
Epoch 30/50
500/500 [==============================] - 2388s 5s/step - loss: 70.3340 - val_loss: 32.6804
Epoch 31/50
500/500 [==============================] - 2406s 5s/step - loss: 67.5038 - val_loss: 36.6804
Epoch 32/50
500/500 [==============================] - 2411s 5s/step - loss: 68.0174 - val_loss: 31.5727

Epoch 00032: ReduceLROnPlateau reducing learning rate to 0.002.
Epoch 33/50
500/500 [==============================] - 2395s 5s/step - loss: 67.3362 - val_loss: 33.4936
Epoch 34/50
500/500 [==============================] - 2402s 5s/step - loss: 65.4268 - val_loss: 33.1911
Epoch 35/50
```

```
500/500 [==============================] - 2399s 5s/step - loss: 65.7545 - val_loss: 33.0299

Epoch 00035: ReduceLROnPlateau reducing learning rate to 0.002.
Epoch 36/50
500/500 [==============================] - 2434s 5s/step - loss: 64.4783 - val_loss: 32.2387
Epoch 37/50
500/500 [==============================] - 2432s 5s/step - loss: 63.4011 - val_loss: 29.4551
Epoch 38/50
500/500 [==============================] - 2441s 5s/step - loss: 62.9743 - val_loss: 28.3006
Epoch 39/50
500/500 [==============================] - 2456s 5s/step - loss: 61.8167 - val_loss: 27.4117
Epoch 40/50
500/500 [==============================] - 2470s 5s/step - loss: 61.1573 - val_loss: 31.0975
Epoch 41/50
500/500 [==============================] - 2392s 5s/step - loss: 60.0332 - val_loss: 30.8048
Epoch 42/50
500/500 [==============================] - 2388s 5s/step - loss: 59.3688 - val_loss: 28.4462

Epoch 00042: ReduceLROnPlateau reducing learning rate to 0.002.
Epoch 43/50
500/500 [==============================] - 2396s 5s/step - loss: 58.6502 - val_loss: 26.9193
Epoch 44/50
500/500 [==============================] - 2396s 5s/step - loss: 57.9148 - val_loss: 30.2132
Epoch 45/50
500/500 [==============================] - 2381s 5s/step - loss: 57.0164 - val_loss: 31.5943
Epoch 46/50
500/500 [==============================] - 2438s 5s/step - loss: 56.2615 - val_loss: 28.4853

Epoch 00046: ReduceLROnPlateau reducing learning rate to 0.002.
Epoch 47/50
500/500 [==============================] - 2508s 5s/step - loss: 56.1670 - val_loss: 30.7610
Epoch 48/50
500/500 [==============================] - 2469s 5s/step - loss: 55.5824 - val_loss: 30.8394
Epoch 49/50
500/500 [==============================] - 2594s 5s/step - loss: 54.3359 - val_loss: 29.4281

Epoch 00049: ReduceLROnPlateau reducing learning rate to 0.002.
Epoch 50/50
500/500 [==============================] - 2725s 5s/step - loss: 53.9117 - val_loss: 30.5559
cropping-2.h5
Epoch 1/50
354/500 [====================>...] - ETA: 11:46 - loss: 83.0287

        ---------------------------------------------------------------------------

        KeyboardInterrupt                         Traceback (most recent call last)
```

```
    <ipython-input-10-f9fc98992254> in <module>()
     12             EarlyStopping(monitor='val_loss', patience=9, min_delta=0.1, verbose=1)
     13             ReduceLROnPlateau(monitor='val_loss', patience=3, min_delta=0.1, factor
---> 14             ModelCheckpoint(model_name, save_best_only=True, save_weights_only=True
     15         ])
     16     model.load_weights(model_name)


    /anaconda3/lib/python3.6/site-packages/keras/legacy/interfaces.py in wrapper(*args, **k
     89                 warnings.warn('Update your `' + object_name + '` call to the ' +
     90                               'Keras 2 API: ' + signature, stacklevel=2)
---> 91             return func(*args, **kwargs)
     92         wrapper._original_function = func
     93         return wrapper


    /anaconda3/lib/python3.6/site-packages/keras/engine/training.py in fit_generator(self,
   1416             use_multiprocessing=use_multiprocessing,
   1417             shuffle=shuffle,
-> 1418             initial_epoch=initial_epoch)
   1419
   1420     @interfaces.legacy_generator_methods_support


    /anaconda3/lib/python3.6/site-packages/keras/engine/training_generator.py in fit_genera
    215                 outs = model.train_on_batch(x, y,
    216                                             sample_weight=sample_weight,
--> 217                                             class_weight=class_weight)
    218
    219                 outs = to_list(outs)


    /anaconda3/lib/python3.6/site-packages/keras/engine/training.py in train_on_batch(self
   1215             ins = x + y + sample_weights
   1216         self._make_train_function()
-> 1217         outputs = self.train_function(ins)
   1218         return unpack_singleton(outputs)
   1219


    /anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py in __call__
   2713                 return self._legacy_call(inputs)
   2714
-> 2715             return self._call(inputs)
   2716         else:
   2717             if py_any(is_tensor(x) for x in inputs):
```

```
/anaconda3/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py in _call(sel
   2673             fetched = self._callable_fn(*array_vals, run_metadata=self.run_metadata
   2674         else:
-> 2675             fetched = self._callable_fn(*array_vals)
   2676         return fetched[:len(self.outputs)]
   2677

/anaconda3/lib/python3.6/site-packages/tensorflow/python/client/session.py in __call__
   1437           ret = tf_session.TF_SessionRunCallable(
   1438               self._session._session, self._handle, args, status,
-> 1439               run_metadata_ptr)
   1440         if run_metadata:
   1441           proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)

KeyboardInterrupt:
```

```
In [14]: model.load_weights('cropping-1.h5')
         loss1 = model.evaluate(val_a, val_b, verbose=0)
         model.load_weights('cropping-2.h5')
         loss2 = model.evaluate(val_a, val_b, verbose=0)
         model.load_weights('cropping-3.h5')
         loss3 = model.evaluate(val_a, val_b, verbose=0)
         model_name = 'cropping-1.h5'
         if loss2 <= loss1 and loss2 < loss3: model_name = 'cropping-2.h5'
         if loss3 <= loss1 and loss3 <= loss2: model_name = 'cropping-3.h5'
         model.load_weights(model_name)
         loss1, loss2, loss3, model_name

Out[14]: (26.919307861328125, 'cropping-1.h5')

In [15]: model2 = build_model(with_dropout=False)
         model2.load_weights(model_name)
         model2.summary()
```

```
_________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=========================================================================================
input_3 (InputLayer)         (None, 128, 128, 1)       0
_________________________________________________________________________________________
conv2d_35 (Conv2D)           (None, 128, 128, 64)      5248        input_3[0][0]
_________________________________________________________________________________________
conv2d_36 (Conv2D)           (None, 128, 128, 64)      36928       conv2d_35[0][0]
_________________________________________________________________________________________
batch_normalization_13 (BatchNo (None, 128, 128, 64)   256         conv2d_36[0][0]
_________________________________________________________________________________________
conv2d_37 (Conv2D)           (None, 64, 64, 64)        16448       batch_normalization_13[0][0]
```

```
conv2d_38 (Conv2D)              (None, 64, 64, 64)   36928       conv2d_37[0][0]
conv2d_39 (Conv2D)              (None, 64, 64, 64)   36928       conv2d_38[0][0]
batch_normalization_14 (BatchNo (None, 64, 64, 64)   256         conv2d_39[0][0]
conv2d_40 (Conv2D)              (None, 32, 32, 64)   16448       batch_normalization_14[0][0]
conv2d_41 (Conv2D)              (None, 32, 32, 64)   36928       conv2d_40[0][0]
conv2d_42 (Conv2D)              (None, 32, 32, 64)   36928       conv2d_41[0][0]
batch_normalization_15 (BatchNo (None, 32, 32, 64)   256         conv2d_42[0][0]
conv2d_43 (Conv2D)              (None, 16, 16, 64)   16448       batch_normalization_15[0][0]
conv2d_44 (Conv2D)              (None, 16, 16, 64)   36928       conv2d_43[0][0]
conv2d_45 (Conv2D)              (None, 16, 16, 64)   36928       conv2d_44[0][0]
batch_normalization_16 (BatchNo (None, 16, 16, 64)   256         conv2d_45[0][0]
conv2d_46 (Conv2D)              (None, 8, 8, 64)     16448       batch_normalization_16[0][0]
conv2d_47 (Conv2D)              (None, 8, 8, 64)     36928       conv2d_46[0][0]
conv2d_48 (Conv2D)              (None, 8, 8, 64)     36928       conv2d_47[0][0]
batch_normalization_17 (BatchNo (None, 8, 8, 64)     256         conv2d_48[0][0]
conv2d_49 (Conv2D)              (None, 4, 4, 64)     16448       batch_normalization_17[0][0]
conv2d_50 (Conv2D)              (None, 4, 4, 64)     36928       conv2d_49[0][0]
conv2d_51 (Conv2D)              (None, 4, 4, 64)     36928       conv2d_50[0][0]
batch_normalization_18 (BatchNo (None, 4, 4, 64)     256         conv2d_51[0][0]
max_pooling2d_5 (MaxPooling2D)  (None, 4, 1, 64)     0           batch_normalization_18[0][0]
max_pooling2d_6 (MaxPooling2D)  (None, 1, 4, 64)     0           batch_normalization_18[0][0]
flatten_5 (Flatten)             (None, 256)          0           max_pooling2d_5[0][0]
flatten_6 (Flatten)             (None, 256)          0           max_pooling2d_6[0][0]
dense_7 (Dense)                 (None, 16)           4112        flatten_5[0][0]
```

```
dense_8 (Dense)                 (None, 16)           4112        flatten_6[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 32)           0           dense_7[0][0]
                                                                 dense_8[0][0]
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 4)            132         concatenate_3[0][0]
==================================================================================================
Total params: 503,588
Trainable params: 502,820
Non-trainable params: 768
__________________________________________________________________________________________________
```

```
In [16]: model2.compile(Adam(lr=0.002), loss='mean_squared_error')
         model2.evaluate(val_a, val_b, verbose=0)
```

```
Out[16]: 26.919307861328125
```

```
In [17]: # Recompute the mean and variance running average without dropout
         for layer in model2.layers:
             if not isinstance(layer, BatchNormalization):
                 layer.trainable = False
         model2.compile(Adam(lr=0.002), loss='mean_squared_error')
         model2.fit_generator(TrainingData(), epochs=1, max_queue_size=12, workers=6, verbose=1
         for layer in model2.layers:
             if not isinstance(layer, BatchNormalization):
                 layer.trainable = True
         model2.compile(Adam(lr=0.002), loss='mean_squared_error')
         model2.save('cropping.model')
```

```
Epoch 1/1
500/500 [==============================] - 1376s 3s/step - loss: 10.1915 - val_loss: 24.3823
```

```
In [18]: model2.evaluate(val_a, val_b, verbose=0)
```

```
Out[18]: 24.38229362487793
```

```
In [19]: images = []
         for i,(p,coords) in enumerate(val[:25]):
             a         = val_a[i:i+1]
             rect1     = val_b[i]
             rect2     = model2.predict(a).squeeze()
             img       = array_to_img(a[0]).convert('RGB')
             draw      = Draw(img)
             draw.rectangle(rect1, outline='red')
             draw.rectangle(rect2, outline='yellow')
             images.append(img)
         show_whale(images)
```